Chapter Six: Contents

(Selectors/Iteration Databases– 31 August 2004– LA-UR 00-1725 – TRANSIMS 3.1)

Chapter Six: Figures

Chapter Six—Selectors/Iteration Databases

1. Introduction

1.1 Overview

A key distinguishing feature of TRANSIMS is the process known as iterative feedback. Feedback provides a natural way to tailor models (of activity locations, mode selections, route planning, etc.) to specific, possibly overlapping, subpopulations. Feedback enables the overall computational system to reflect "learned" behavior within the simulated population represented. Feedback involves two crucial processes:

- Biased selection – defining a subpopulation based on any static or dynamic information about travelers available to TRANSIMS.

- Updating travelers – revising the selected subpopulation's use of the transportation system by controlling the quality of information about the system available to them.

The information about travelers available to TRANSIMS consists of the traveler-specific data contained in population, activity, plan, vehicle, and simulation output files. These data are all generated by TRANSIMS under specific hypotheses about the transportation network. By carefully controlling the hypotheses, TRANSIMS can be used to steer travelers toward certain choices.

The mechanics of controlling information flow among TRANSIMS modules is discussed in the Input/Output section of each module's description. This chapter describes the Selector/Iteration Database and how it works together with an iteration script to control the overall TRANSIMS Framework. A typical TRANSIMS study involves repeated iteration between modules. There is no single, "standard" iteration script because different study designs involve different iteration schemes.

One important example of feedback is in solving the traffic assignment problem. The simplest version of this uses a loop between the Route Planner and Traffic Microsimulator modules. On the first pass of the Route Planner, routes are chosen under the hypothesis that travel time is well represented by free speeds on the network (i.e., that travelers do not interact). Correction for traveler interactions can be applied simply by making available to the Route Planner information about actual travel times produced by the Traffic Microsimulator[1]. With this information, the Route Planner will choose different routes for most travelers, resulting in different travel times. In this case, updating travelers is accomplished by re-running the Route Planner with an updated travel time table. However, there is still a wide range of different feedback schemes for this problem which depend on the selection step—exactly which travelers are to be run through the Route Planner with updated travel time information. One selection process is to choose a certain fraction of travelers uniformly at random. The Selector/Iteration

[1] Notice that there is no requirement to provide *correct* travel time information—it might be noisy, or averaged together with travel times used in the previous run.

Database described below supports much more sophisticated processes, though. For example, one could select only travelers with automobile drives of an hour or more who cross a geographic feature (like a river).

Of course, there are many more information flows in TRANSIMS than just the travel time table. Every TRANSIMS module can be used to update only a selected subpopulation using information provided by the Framework. In effect, this is like providing a separate model for every conceivable subdivision of the population without the need for fitting each model separately. For example, work location is chosen using a single simple model for the entire population. If people who commute by bus across a river are assigned work locations poorly, selecting that subpopulation and running the work location assignment model with slightly different input information can change the poorly selected locations for that subpopulation with no change in the model itself.

Notice that a single traveler might be in *two* subpopulations – for example, the previous subpopulation and the subpopulation assigned to households larger than five people who also have longer than average commutes—but no sophisticated correlation structure needs to be built into the model to handle such cases correctly.

Selection is based on both absolute criteria such as traveler's mode, and on relative criteria such as the duration of a trip compared to the duration of all other trips in the subpopulation picked out by the absolute criteria. The relative criteria act as user-specified cost functions. Thus, we might select the 10% of travelers meeting some absolute criteria who have the longest actual travel time compared with their expected travel time.

Fig. 1 gives an indication of how the selection process works. Here data is collected on the travelers' incomes, their travel modes, the length of trips, whether they cross the river, and the relative length of the trip. All travelers with some collection of these characteristics, for example those on bus trips with income >$40k, are collected and the distribution of relative trip duration is formed. A portion of these travelers with the largest duration is selected to travel by a different mode.

Selection and Feedback

The iteration database:

Traveler	Income	Mode	>1	Cross	Relative	. . .
			hour?	river?	duration	
291362	$25K	bus	no	yes	1.2	. . .
291363	$34K	car	yes	no	1.6	. . .
291364	$42K	car	no	yes	1.1	. . .
291365	$ 0K	walk	no	no	1.0	. . .
291366	$38K	car	yes	yes	2.3	. . .
291367	$45K	bus	yes	no	1.4	. . .
291368	$30K	car	yes	yes	1.3	. . .

Selection criterion:	Selects travelers:
bus trips with income >$40K	291367
short trips crossing the river	291362 291364
long car trips not crossing the river, relative duration > 1.3	291363

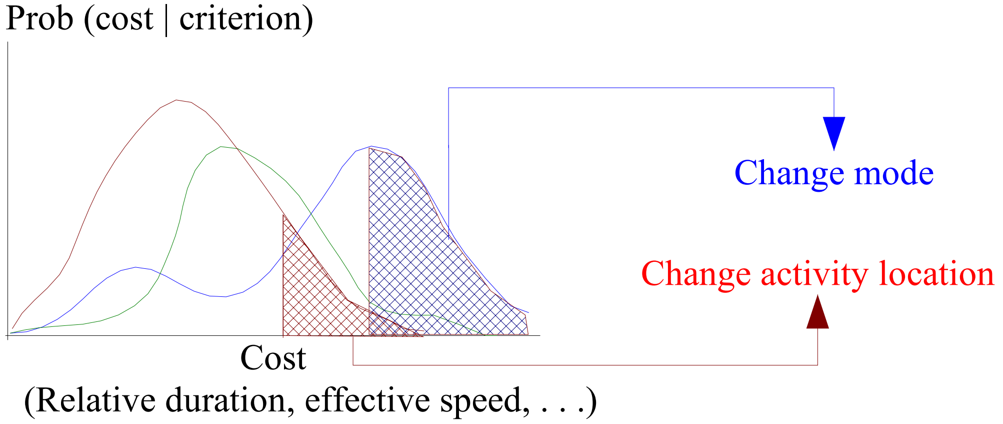


Fig. 1. The selection process.

1.2 The Iterative Process

During each iteration, the user invokes the Collator, Stratifier, and Selector to do the following:

- Read information about the travelers from the Iteration Database.
- Examine each traveler and decide whether to
  - regenerate his or her activities using the Activity Generator,
  - select a new route between his or her existing activities using the Route Planner, or
  - retain his or her existing activities and the planned route between them.
- Write the selections made for each traveler into data files that can be read by the Activity Generator and the Route Planner when they are executed.

Fig. 2 and Fig. 3 illustrate a step in the iterative process.

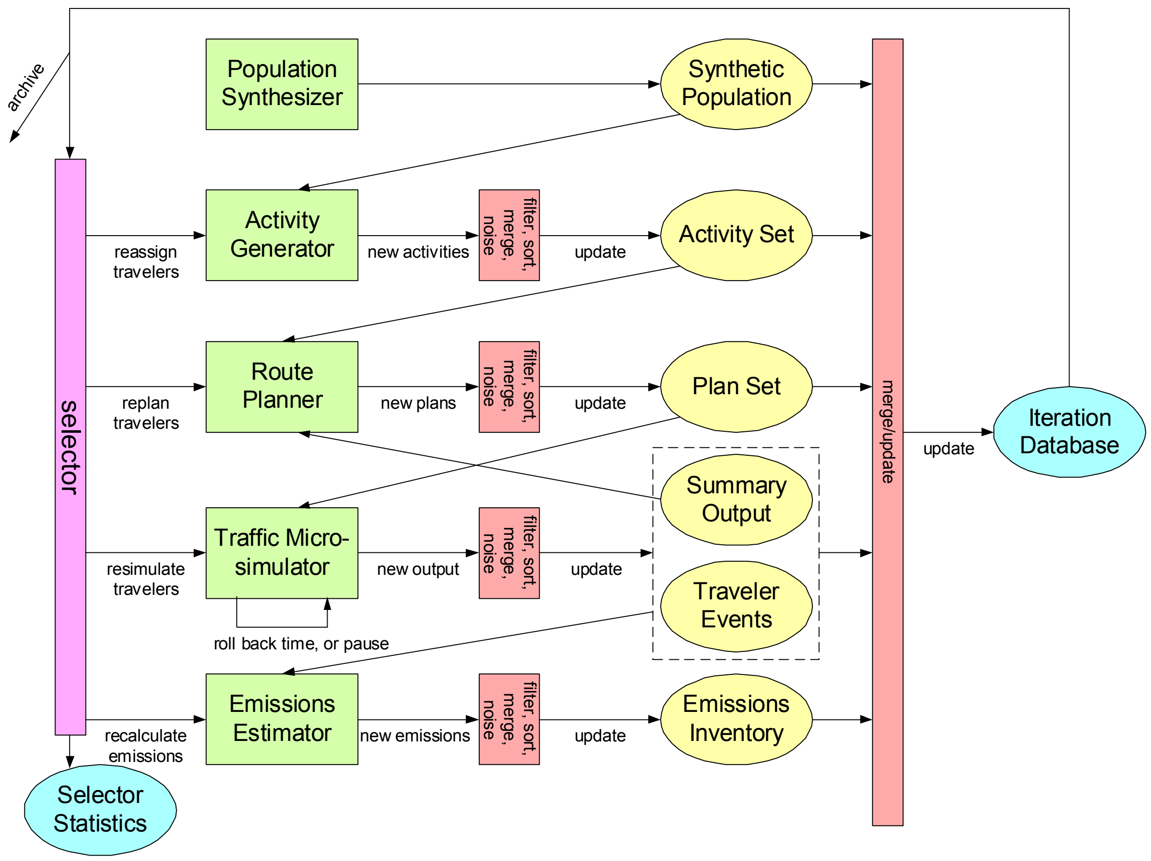


Fig. 2. Location of the Selector/Iteration Database within a typical TRANSIMS experimental design.

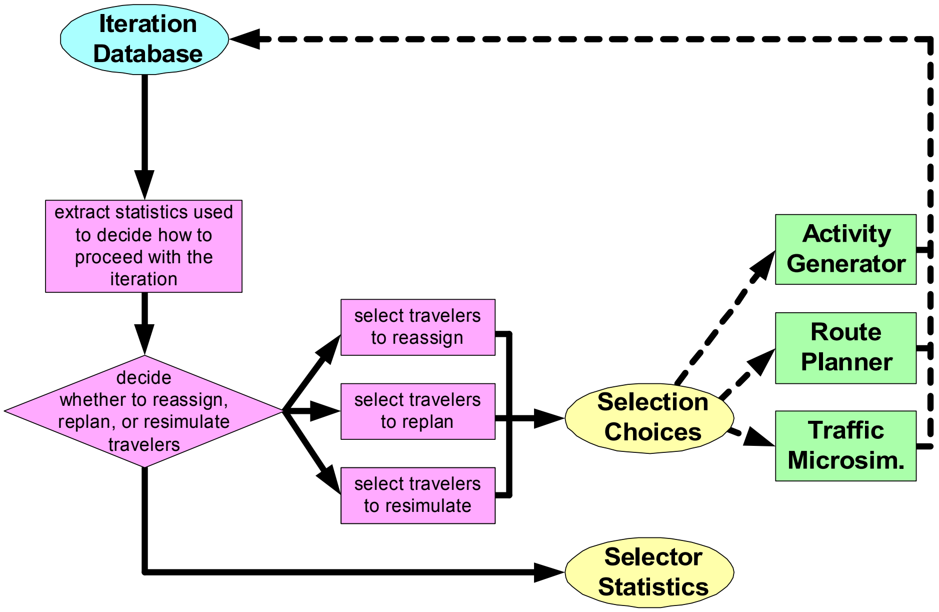

Iteration Database
extract statistics used to decide how to proceed with the iteration
decide whether to reassign, replan, or resimulate travelers
select travelers to reassign
select travelers to replan
select travelers to resimulate
Selection Choices
Activity Generator
Route Planner
Traffic Microsim.
Selector Statistics

Fig. 3. Typical Selector/Iteration Database logic.

After the Selector completes the selection process for all travelers, the Activity Generator, Route Planner, or Traffic Microsimulator runs to calculate the updated activity set, plan set, or microsimulation output files, respectively (according to the decisions made by the Selector).

The iteration script will reinvoke a Selector again at the start of the next iteration in the study. Fig. 4 shows examples of four possible progressions, as determined by the Selector.

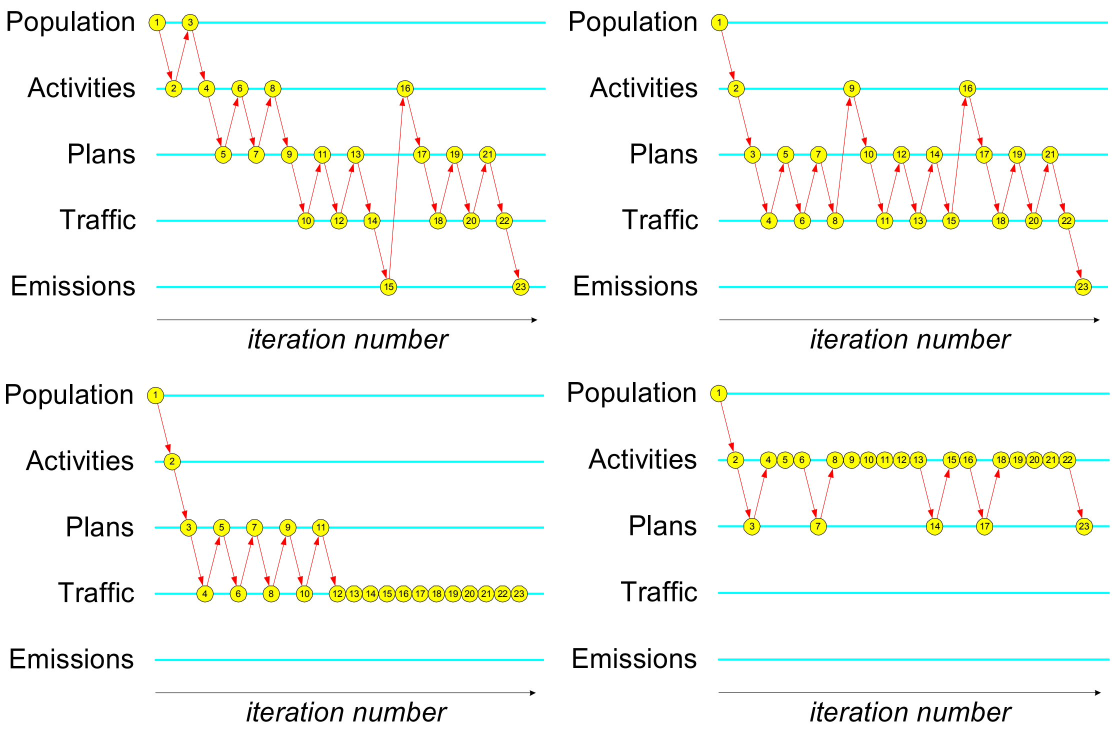


Fig. 4. Four examples of iteration progressions.

The Iteration Database is the archive of information about travelers across iterations. The Selector uses this information to make its selection decisions. The data contained in the database are chosen by the user from:

- The fields of the population, activity, and plan files—for example, income, mode preference, or the expected duration of a trip.
- Information extracted from detailed Traffic Microsimulator event output—for example, the actual duration of a trip.
- Information deduced from combinations of the previous two—for example the duration of a trip relative to its expected duration.

The left side of Fig. 5 shows this data flow into the Selector/Iteration Database.

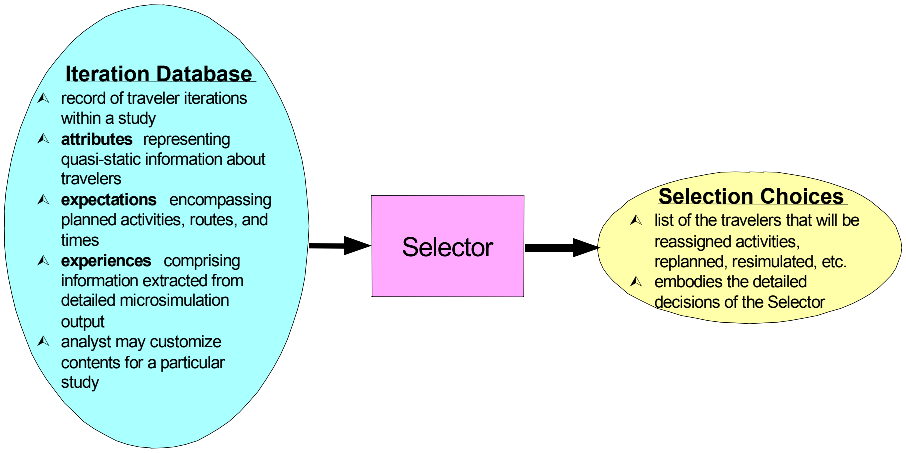

Iteration Database
record of traveler iterations within a study
attributes representing quasi-static information about travelers
expectations encompassing planned activities, routes, and times
experiences comprising information extracted from detailed microsimulation output
analyst may customize contents for a particular study
Selector
Selection Choices
list of the travelers that will be reassigned activities, replanned, resimulated, etc.
embodies the detailed decisions of the Selector

Fig. 5. Typical Selector/Iteration Database data flow.

1.3 TRANSIMS Framework Flexibility

The Framework's flexibility allows for countless variations in the iteration process. For example, in some studies, the Selector may run after the Activity Generator or Route Planner completes its execution. Thus, the Selector can decide which of the generated activities or plans will be accepted for travelers. Those not accepted are discarded and new activities or plans are produced.

The iteration script has the potential to make additional choices, such as the following:

- which version of the Activity Generator, Route Planner, or Traffic Microsimulator will run during the present iteration;
- if transit schedules will be adjusted or vehicles added or removed from the transit fleet;
- if network characteristics (such as traffic signal timing, congestion pricing, or roadway information signs) will be altered;
- which travelers receive data from traffic information systems; or
- whether to complete the study (i.e., end the iteration) because the iterations have converged sufficiently (or diverged).

Several implementations have been written that have use in typical transportation planning studies. For example, Fig. 3 shows a typical iteration scheme that is set up by the Selector/Iteration Database script. In this scheme, activities, plans, and microsimulations are iterated until traffic behavior on the network stabilizes. It is not difficult for analysts to write additional iteration scripts for their own specialized studies.

2. Modeling Tools

TRANSIMS uses selective feedback for modeling. Actions such as running the Activity Regenerator or choosing a new route are performed on a selected set of activities or households. TRANSIMS provides three tools for choosing the sets to be acted on:

- Collator
- Stratifier
- Selector

Each of these tools is described in detail below. Briefly, the Collator gathers input and output data for each trip and creates a database. The Stratifier allows the user to discretize or bin values in any column of the database and create an arbitrary number of multi-way tables over any binned values. The Selector chooses a cell from any of the multi-way tables, possibly based on a cost function, and selects a subset of travelers or trips from that cell for feedback. The Collator provides a means of translating data in TRANSIMS-format files into a simple, but possibly large, ASCII, comma-separated database. It frees the user from the responsibility of parsing all of the different formats and bringing together data from many different sources. The Stratifier and Selector, on the other hand, implement a rudimentary database query and reporting system that the user may replace with other more familiar systems.

We provide two running examples below.

- The first illustrates how one might select short trips that cross a river and assign new destinations for the trips.
- The second shows how to find trips made by people over 17 years old for which the expected time in an auto is much shorter than expected time walking, and reroute them.

The complete configuration file used to create the databases for these examples is included in the Appendix H.

2.1 Collator

The Collator can extract data from all of the most important TRANSIMS input or output files: Network, Population, Activity, Plan, Event, Transit Route, and Activity Generator or Route Planner problem files. It can calculate values based on data from several different sources: for example, it uses the activity location table together with an activity file to calculate Euclidean distance between activities.

The Collator creates a database with one record for each trip made by each traveler from every household specified in its household file, or in the population file, if no household file is specified. Each record contains a few pre-determined fields and as many other fields as the user has requested. Fields that are present in every record are: `household id`, `traveler id`, `trip id` (as it would appear in the traveler's plan), `starting` and `ending activity ids` (as they appear in the activity file), and `tour` and `subtour id`. The `tour` and `subtour` are not used in other TRANSIMS modules, but are provided for the user's convenience. They are defined by the following algorithm:

- Each trip starting from the home location begins a new tour.
- Each trip from an anchor activity that returns to the same anchor location before returning to the home location begins a new subtour.

The user requests fields by turning on any of the configuration file keys shown in the Appendixes of this document.

If the data for a given field is not available—for example, if there is no plan corresponding to a particular trip—that column will be blank in the output. If the data cannot be calculated—for example, if a denominator is zero—an "NA" will appear in that column.

The Collator output can be indexed if desired. If the user requests an index, results from previous runs of the Collator will be made available to the current invocation. If the underlying data files have not changed since the most recent Collator run or if data is missing from the data files, the most recent value from a previous database will be used. If an index is present, certain fields can be added to the database, such as a number indicating the last iteration on which a variable changed. Indexing can be time-consuming, and is not often necessary.

2.1.1 Collator Example

For the first task, we will need to know the expected duration of trips crossing a river. First, we create a set of polygons that distinguish the two sides of a river. These are stored in the file `$TRANSIMS_ROOT/network/rivers.polygons`, called a "User Analysis Zone" or UAZ file by the Collator. In our case, there are two rivers that merge and thus three polygons. To find out whether a trip's origin and destination are in different polygons, we add the following keys to a configuration file:

```
  SEL_UAZ_FILE_1           $TRANSIMS_ROOT/network/rivers.polygons
  SEL_USE_CROSS_BOUNDARY   1
```

If we wanted to be more specific about which polygonal region the trip started or ended in, we could also add:

```
  SEL_USE_START_REGION     1
  SEL_USE_END_REGION       1
```

Finally, if we are only interested in trips that start and end in specific polygons, say starting in polygon 1 of UAZ file 1 and ending in polygon 2 or 3 of UAZ file 1, we could use the following configuration file keys:

```
SEL_USE_START_IN_REGION    1, 1
SEL_USE_END_IN_REGION      1, 2; 1, 3
SEL_USE_AND                START_IN_REGION_1_1,
END_IN_REGION_1_2; START_IN_REGION_1_1, END_IN_REGION_1_3
```

The SEL_USE_AND configuration file key creates two fields, each of which is the logical AND of its arguments. In this case, the first field is true if, and only if, the trip starts in polygon 1 of UAZ file 1 and ends in polygon 2 of UAZ file 1.

In this example, we have included all of these different REGION and BOUNDARY configuration file keys.

Only the START_IN_REGION, END_IN_REGION, and AND configuration file keys are required for what follows.

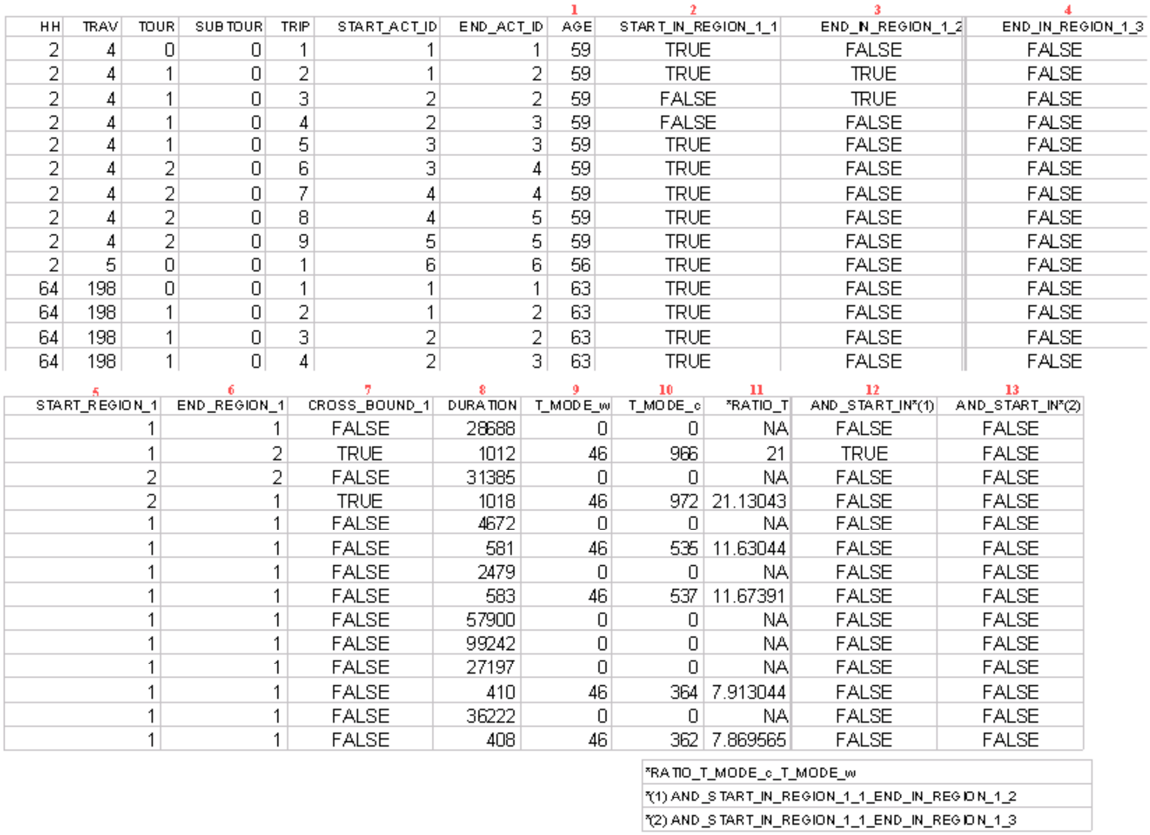
To find the expected travel time (from the Plan file), we add the following configuration file key:

```
  SEL_USE_DURATION   1.
```

The second task will require the age of the traveler and the ratio of expected travel times in modes ‘w' and 'c':

```
  SEL_USE_AGE      1
  SEL_USE_T_MODE   w; c
  SEL_USE_RATIO    T_MODE_c, T_MODE_w
```

Running the Collator with these configuration file keys on 10,265 households out of a full Portland Activity and Plan set takes about 2.5 hours with a 400 MHz Sun SPARC processor. The Collator can be run in a distributed fashion, with each processor handling a different set of households. The resulting 20 Megabyte database contains 258,000 records with 20 fields per record. Fig. 6 shows a sample of the Collator output database for these configuration file keys.



							1	2	3	4
HH	TRAV	TOUR	SUBTOUR	TRIP	START_ACT_ID	END_ACT_ID	AGE	START_IN_REGION_1_1	END_N_REGION_1_2	END_IN_REGION_1_3
2	4	0	0	1	1	1	59	TRUE	FALSE	FALSE
2	4	1	0	2	1	2	59	TRUE	TRUE	FALSE
2	4	1	0	3	2	2	59	FALSE	TRUE	FALSE
2	4	1	0	4	2	3	59	FALSE	FALSE	FALSE
2	4	1	0	5	3	3	59	TRUE	FALSE	FALSE
2	4	2	0	6	3	4	59	TRUE	FALSE	FALSE
2	4	2	0	7	4	4	59	TRUE	FALSE	FALSE
2	4	2	0	8	4	5	59	TRUE	FALSE	FALSE
2	4	2	0	9	5	5	59	TRUE	FALSE	FALSE
2	5	0	0	1	6	6	56	TRUE	FALSE	FALSE
64	198	0	0	1	1	1	63	TRUE	FALSE	FALSE
64	198	1	0	2	1	2	63	TRUE	FALSE	FALSE
64	198	1	0	3	2	2	63	TRUE	FALSE	FALSE
64	198	1	0	4	2	3	63	TRUE	FALSE	FALSE

5	6	7	8	9	10	11	12	13
START_REGION_1	END_REGION_1	CROSS_BOUND_1	DURATION	T_MODE_w	T_MODE_c	*RATIO_T	AND_START_IN*(1)	AND_START_IN*(2)
1	1	FALSE	28688	0	0	NA	FALSE	FALSE
1	2	TRUE	1012	46	966	21	TRUE	FALSE
2	2	FALSE	31385	0	0	NA	FALSE	FALSE
2	1	TRUE	1018	46	972	21.13043	FALSE	FALSE
1	1	FALSE	4672	0	0	NA	FALSE	FALSE
1	1	FALSE	581	46	535	11.63044	FALSE	FALSE
1	1	FALSE	2479	0	0	NA	FALSE	FALSE
1	1	FALSE	583	46	537	11.67391	FALSE	FALSE
1	1	FALSE	57900	0	0	NA	FALSE	FALSE
1	1	FALSE	99242	0	0	NA	FALSE	FALSE
1	1	FALSE	27197	0	0	NA	FALSE	FALSE
1	1	FALSE	410	46	364	7.913044	FALSE	FALSE
1	1	FALSE	36222	0	0	NA	FALSE	FALSE
1	1	FALSE	408	46	362	7.869565	FALSE	FALSE

*RATIO_T_MODE_c_T_MODE_w
*(1) AND_START_IN_REGION_1_1_END_IN_REGION_1_2
*(2) AND_START_IN_REGION_1_1_END_IN_REGION_1_3

Fig. 6. Example Iteration Database.

Notice that "NA" appears for the value of a field when it requires division by 0. In general, inappropriate fields are either left blank or given an "NA". Records with blank or "NA" values are not used in Stratifier or Selector calculations that rely on that field. At this point, if the user wished to analyze the data by tour rather than by trip, s/he could use text processing tools to aggregate the data and prepare a new file for later use by the Stratifier and Selector. The only requirements imposed by the software are:

- two lines of header information, the second of which lists all the field names
- fields are comma-separated ASCII text;
- the fields `HH`, `TRAV`, `TRIP`, `START_ACT_ID`, and `END_ACT_ID` are required.

2.2 Stratifier

The Stratifier has two tasks: discretize variables and combine them into multi-way tables. Each discretized variable is known as a "binning", and each binning is given a numeric identifier.

Discretization is accomplished in any of the following three ways;

- automatically (best for fields that already contain discrete data with only a few different values). Each different value is assigned a numeric bin id.
- manually, specifying the number of bins. The data in a field is sorted and placed into the user-specified number of quantiles. If the data is discrete and heavily concentrated on a few values, the algorithm may reduce the number of quantiles.
- manually, specifying the bin boundaries.

The user chooses one of these methods for each binning using configuration file keys. The boundaries for each binning are listed in the header of the Stratifier's output database.

The user then instructs the Stratifier to create a set of k-way tables from the binnings. Each binning can be used in any number of different tables. Each table is called a "stratification", and each stratification is given a numeric identifier. Furthermore, each cell in the k-way table is given an index. For example, the user could choose binnings 1 and 3 to create a 2-way stratification. If binning 1 contained two distinct bins and binning 3 contained 3, the 2-way table would contain a total of 6 cells, as illustrated below.

```
         0            1
------------------------------
0 | (0, 0) -> 0 | (0, 1) -> 1
  |             |
1 | (1, 0) -> 2 | (1, 1) -> 3
  |             |
2 | (2, 0) -> 4 | (2, 1) -> 5
```

The Stratifier's output database contains one record for each record in the Collator's database. Each record contains the pre-determined fields `household id`, `traveler id`, `trip`, and starting and ending activity for the trip. It also contains one field for each binning specified and one field for each stratification. The stratification field's value is the index of the cell into which this record falls.

2.2.1 Stratifier Example

For the first task, we want to find all trips that start in region 1 and end in region 2 or 3, with a duration of 20 minutes or less. We will add binnings that address each of these categories. First, we specify a binning for the `DURATION` field into two bins with a boundary of 20 minutes (= 1200 seconds):

```
  SEL_BIN_NAME_0      duration_bin
  SEL_BIN_FIELD_0     DURATION
  SEL_BIN_BOUNDS_0    1200
```

Next, we create separate binnings for trips that start in region 1 and end in region 2 or 3. Since these are Boolean fields, we need not specify bin boundaries.

```
  SEL_BIN_NAME_1      1_to_2
  SEL_BIN_FIELD_1     AND_START_IN_REGION_1_1_END_IN_REGION_1_2

  SEL_BIN_NAME_2      1_to_3
  SEL_BIN_FIELD_2     AND_START_IN_REGION_1_1_END_IN_REGION_1_3
```

For the second task, we will bin by age:

```
  SEL_BIN_NAME_3      age_bin
  SEL_BIN_FIELD_3     AGE
  SEL_BIN_BOUNDS_3    18
```

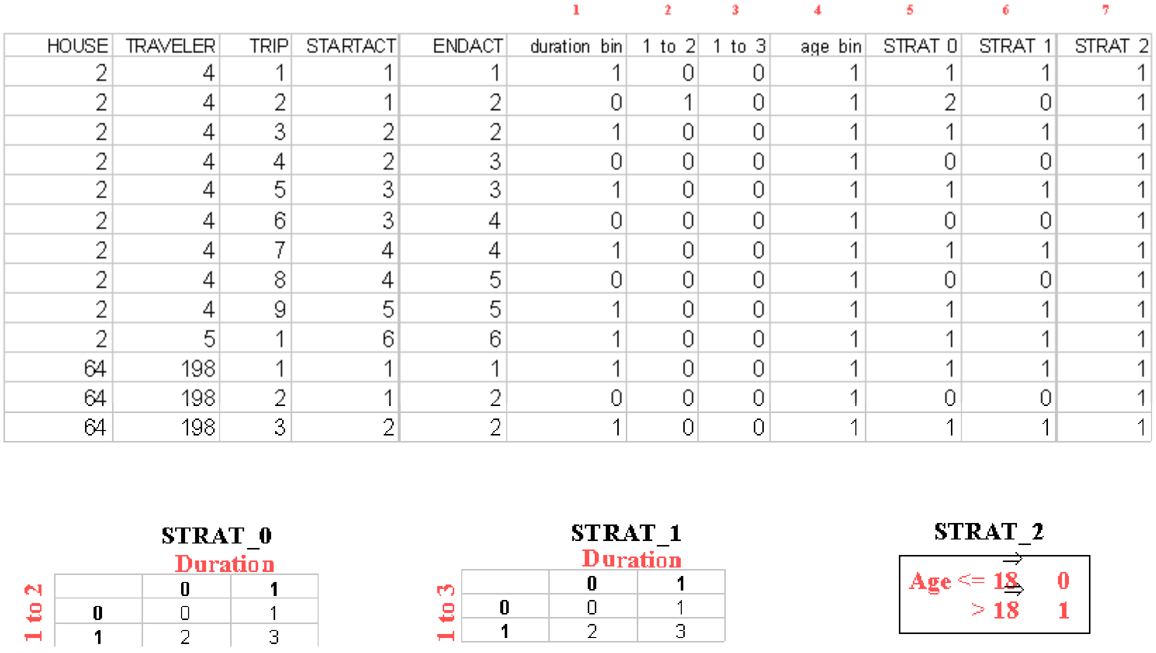
Finally, we create the k-way tables, or stratifications, we will use in the selection process.

For the first task, we will construct a pair of two-way tables: one from binnings 0 and 1; and one from binnings 0 and 2.

For the second task, we will use the age binning (number 3) by itself. The configuration file key `SEL_STRAT_BINS` specifies this arrangement:

```
  SEL_STRAT_BINS     0, 1; 0, 2; 3
```

The Stratifier takes less than two minutes to run on the 258,000 record database created by the Collator above. Fig. 7 shows a sample of the Stratifier output database for the same trips shown in the Collator database sample above.



					1	2	3	4	5	6	7
HOUSE	TRAVELER	TRIP	STARTACT	ENDACT	duration bin	1 to 2	1 to 3	age bin	STRAT 0	STRAT 1	STRAT 2
2	4	1	1	1	1	0	0	1	1	1	1
2	4	2	1	2	0	1	0	1	2	0	1
2	4	3	2	2	1	0	0	1	1	1	1
2	4	4	2	3	0	0	0	1	0	0	1
2	4	5	3	3	1	0	0	1	1	1	1
2	4	6	3	4	0	0	0	1	0	0	1
2	4	7	4	4	1	0	0	1	1	1	1
2	4	8	4	5	0	0	0	1	0	0	1
2	4	9	5	5	1	0	0	1	1	1	1
2	5	1	6	6	1	0	0	1	1	1	1
64	198	1	1	1	1	0	0	1	1	1	1
64	198	2	1	2	0	0	0	1	0	0	1
64	198	3	2	2	1	0	0	1	1	1	1

STRAT_0

		Duration	
		0	1
1 to 2	0	0	1
	1	2	3

STRAT_1

		Duration	
		0	1
1 to 3	0	0	1
	1	2	3

STRAT_2

Age <= 18 →	0
> 18 →	1

Fig. 7. Example Stratifier Database.

The first line of the file (not show here) describes each of the binnings. There are three different types: rational, categorical, and ordinal. In a "rational" binning the elements are assumed to be floating-point numbers, and bin bounds are as indicated; in a "categorical" binning, the elements take on one of the few indicated discrete values, each of which is one bin; in an "ordinal" binning, they take on all integer values between the indicated upper and lower bounds.

The Stratifier adds the same `household`, `traveler`, `trip`, and `start and end activity` fields as the Collator. It does not add `tour` or `subtour` information.

2.3 Selector

The Selector chooses cells from a stratification and selects a subset of travelers or trips within the cell. The cells can be picked by index (using the `PICK_CELL` selection algorithm). The index of a desired cell can be determined as follows:

> If the stratification is a k-way table, and the number of bins in each of the k binnings making up the table is n(k), and the desired cell is made up of bin i(k) of each of the binnings, its index is
>
> $$\sum_{j=1}^{k}\{i(j)\prod_{m-0}^{j-1}n(m)\}$$
>
> where n(0) = 1.

In the formula above, the bin indexes i(k) are in the range [0, n(k) - 1].

It is probably worthwhile for the user to verify that the cell index used in fact corresponds to the desired one by looking at the values of variables of interest for a traveler assigned to that cell.

That is, the user can identify a trip to be selected by examining the Collator's database, and can determine which cell that trip falls in by examining the Stratifier's database.

More generally, each entry in a stratification can be associated with a cost function, and a cell can be picked automatically based on the distribution of costs for that cell. The cost function can be any field in the Collator's output database. For example, if the user selects travel time as a cost function, s/he can pick the cell that has the highest or lowest mean travel time; or the cell that has the largest or smallest standard deviation or range in travel times.

Once a cell has been chosen, the Selector will pick some of the trips within that cell. The user can specify that all elements of the cell be chosen, or a subset chosen uniformly at random, or the elements in the high or low tails of the cost function distribution.

A single run of the Selector can be used to select many different subsets of trips or travelers—each stratification in the Stratifier's output database can be associated with a cost function one or more times.

There are two possible goals for selection. One is to identify activities to be changed using the Activity Regenerator. The other is to identify households to be re-routed. The Activity Regenerator expects a file of regenerator commands (see TRANSIMS V3.0, Volume 3 (*Modules*), Chapter 3 (*Activity Generator*), Section 5 (*Activity Regenerator*)), each of which includes a `traveler id` and an `activity id`. In addition, it is likely that each household for which an activity has been changed should be replanned. The Route Planner expects a file of household IDs. For each selection, the user specifies a "goal". The selected travelers' household IDs and activity IDs are added to an Activity Regenerator file, along with the corresponding character string from the `GOAL` configuration file key. In addition, the household ID is added to a Route Planner feedback file.

Each selection creates a different pair of Activity Regenerator and Route Planner feedback files. The Route Planner feedback files should be concatenated, sorted, and duplicate lines removed so that each household appears only once. The Activity Regenerator files should be concatenated so that all activities for a given household can be updated simultaneously. The order in which commands are read by the Activity Regenerator is significant. The user should be careful to concatenate Activity Regenerator files in the order s/he wishes the Regenerator commands to be applied.

2.3.1 Selector Example

The following example illustrates one way of performing the desired selections. It is not unique.

The first task asks for all the elements of a particular cell in the first stratification. We would like to use the cell corresponding to bin 1 of the first binning (which has a total of 2 bins) and bin 1 of the second binning, which, according to the formula above, gives a cell index of:

$1 + 1 * 2 = 3$

In practice, we would create a similar selection from the second stratification. We don't use this selection in the following example because it does not illustrate anything different from the first selection.

Even though no cost function is required for the selection, the current implementation of the Selector requires one to be provided. It is best if the cost function not have any "NA" values in it, because that will unnecessarily remove some records from consideration. Hence we choose HH, which is guaranteed to be defined for every record. For the second task, we will again pick a specific cell, but not all of the elements of that cell are needed. In this case, we are using bin 1 of the only binning, so the cell index is also 1. We use the ratio of travel times in walk and car mode as the cost function:

```
SEL_USE_STRATIFICATION   0; 2
SEL_ALGORITHM            PICK_CELL 3; PICK_CELL 1
SEL_COST                 HH; RATIO_T_MODE_c_T_MODE_w
SEL_BIN_SEL_ALGO         ALL; TAIL, 1, , 1.5
```

The parameters to the TAIL" within-cell selection algorithm specify that the elements with costs above 1.5 are to be chosen.

Finally, we want to assign a goal for each selection. The goal of the first task is to relocate the ending activity for each selected trip. The goal of the second task is to reroute.

```
SEL_GOAL      LS 0.01; REROUTE
```

The output of a Selector run with this configuration file will consist of four files. There will be one Activity Feedback file and one Router Feedback file for each of the two selections. The Activity Feedback file for selection 2 should be ignored because we intend only to re-route these households.

If several different Activity Regenerator commands were generated, the user should concatenate the files in the order s/he wishes the commands to be applied. For example, it makes little sense to change an activity location after a household's activities have been regenerated from the survey, so files containing a "regenerate activities from the survey" command should be appended after all other files. Similarly, the Router Feedback household files should all be concatenated and sorted and duplicate lines should be removed.

The base filenames for Selector output are specified by two configuration file keys:

```
ACT_FEEDBACK_FILE        /home/eubank/test/feedback.act
ROUTER_HOUSEHOLD_FILE    /home/eubank/test/feedback.router
```

The actual filenames will have "_<n>" appended to them, where "<n>" is

an integer indicating which selection they represent.

The Selector takes less than a minute to run on the 258,000 record database created by the Stratifier above. Here is the first line of the first Activity Regenerator Feedback file:

```
2 2 LS 0.01
```

And here is the first line of the second Router Feedback file:

```
2
```

Appendix A: Iteration Database General Configuration File Keys

Configuration File Key	Description
ROUTER_IGNORABLE_PRIORITIES	See the Route Planner documentation.
ACT_HOME_ACTIVITY_TYPE	The number of the home activity type (non-negative integer). This key should be specified if an activity file is specified.
ACT_SCHOOL_ACTIVITY_TYPE	The number of the school activity type (non-negative integer). This key should be specified if an activity file is specified.
ACT_WORK_ACTIVITY_TYPE	The number of the work activity (non-negative integer). This key should be specified if an activity file is specified.
NET_ACTIVITY_LOCATION_TABLE	The activity location table name. This key is required.
NET_DIRECTORY	The directory where the network files reside. This key is required.
NET_LINK_TABLE	The link table name. This key is required.
NET_NODE_TABLE	The node table name. This key is required.
NET_PARKING_TABLE	The parking table name. This key is required.
NET_PROCESS_LINK_TABLE	The process link table name. This key is required.
NET_TRANSIT_STOP_TABLE	The transit stop table name. This key is required.
SEL_ACTIVITY_FILE SEL_EVENT_FILE	The activity file for use by the Collator. If not present, the ACTIVITY_FILE configuration file key is used. Either SEL_ACTIVITY_FILE or ACTIVITY_FILE must be specified. The event file for use by the Collator.
SEL_ITDB_FILE	The full pathname of the output Iteration Database file generated by the Selector Collator. An iteration number extension is automatically added to the end of this name.
SEL_MESSAGE_LEVEL	Sets the message level for the Collator, Stratifier, and Selector modules. Message levels range from –1 to 4 with the higher numbers reporting more.
SEL_PLAN_FILE	The plan file for use by the Collator. If not present, the PLAN_FILE configuration file key is used.
SEL_POPULATION_FILE	The population file for use by the Collator. If not present, the ACT_POPULATION_FILE configuration file key is used. Either SEL_POPULATION_FILE or ACT_POPULATION_FILE must be specified.
SEL_STRAT_OUT_FILE	The full pathname for the output Iteration Database created by the Stratifier. There may be several of these for each Collator run. Default = strat

Configuration File Key	Description
`SEL_UAZ_FILE_n`	The full pathname of the polygon file specifying User Analysis Zone(s) `n`, where `n` is an integer starting at 1.

Appendix B: Iteration Database Activity Configuration File Keys

Configuration File Key	Description
`SEL_USE_ACT_HH_ID`	If set, directs the Collator to add the household ID from the activity file for the ending activity for the trip to the output Iteration Database
`SEL_USE_ACT_LAST_OK_ITER`	If set, directs the Collator to add a field containing the number of the last iteration on which no problems were reported for this household in the Activity Generator or Regenerator's Problem File. A value of `-1` indicates there has been a problem reported on every iteration.
`SEL_USE_ACT_PERSON_ID`	If set, directs the Collator to add the person ID from the activity file for the ending activity for the trip to the output Iteration Database
`SEL_USE_END_ACT_GROUP_NUM`	If set, directs the Collator to add the activity group number from the activity file for the ending activity for the trip to the output Iteration Database
`SEL_USE_END_ACT_ID`	If set, directs the Collator to add the activity ID from the activity file for the starting activity for the trip to the output Iteration Database
`SEL_USE_END_ACT_LOCATION`	If set, directs the Collator to add the first of the possible locations from the activity file for the ending activity for the trip to the output Iteration Database
`SEL_USE_END_ACT_TYPE`	If set, directs the Collator to add the activity type from the activity file for the ending activity for the trip to the output Iteration Database
`SEL_USE_END_DUR_LB`	If set, directs the Collator to add the duration lower bound from the activity file for the ending activity for the trip to the output Iteration Database. The value is converted from fractional hours to seconds.
`SEL_USE_END_DUR_UB`	If set, directs the Collator to add the duration upper bound from the activity file for the ending activity for the trip to the output Iteration Database. The value is converted from fractional hours to seconds.
`SEL_USE_END_TIME_LB`	If set, directs the Collator to add the end time lower bound from the activity file for the ending activity for the trip to the output Iteration Database.
`SEL_USE_END_TIME_UB`	If set, directs the Collator to add the end time upper bound from the activity file for the ending activity for the trip to the output Iteration Database.
`SEL_USE_END_MODE_PREF`	If set, directs the Collator to add the mode preference from the activity file for the ending activity for the trip to the output Iteration Database

Configuration File Key	Description
SEL_USE_END_OTHER_PARTICIPANTS	If set, directs the Collator to add the number of other participants from the activity file for the ending activity for the trip to the output Iteration Database
SEL_USE_END_POSS_LOC	If set, directs the Collator to add the number of possible locations from the activity file for the ending activity for the trip to the output Iteration Database
SEL_USE_END_PRIORITY	If set, directs the Collator to add the activity priority from the activity file for the ending activity for the trip to the output Iteration Database
SEL_USE_END_START_TIME_LB	If set, directs the Collator to add the start time lower bound from the activity file for the ending activity for the trip to the output Iteration Database
SEL_USE_END_START_TIME_UB	If set, directs the Collator to add the start time upper bound from the activity file for the ending activity for the trip to the output Iteration Database
SEL_USE_END_VEHICLE_ID	If set, directs the Collator to add the vehicle ID from the activity file for the ending activity for the trip to the output Iteration Database
SEL_USE_START_ACT_GROUP_NUM	If set, directs the Collator to add the activity group number from the activity file for the starting activity for the trip to the output Iteration Database.
SEL_USE_START_ACT_ID	If set, directs the Collator to add the activity ID from the activity file for the starting activity for the trip to the output Iteration Database.
SEL_USE_START_ACT_LOCATION	If set, directs the Collator to add the first of the possible locations from the activity file for the starting activity for the trip to the output Iteration Database.
SEL_USE_START_ACT_TYPE	If set, directs the Collator to add the activity type from the activity file for the starting activity for the trip to the output Iteration Database.
SEL_USE_START_DUR_LB	If set, directs the Collator to add the duration lower bound from the activity file for the starting activity for the trip to the output Iteration Database. The value is converted from fractional hours to seconds.
SEL_USE_START_DUR_UB	If set, directs the Collator to add the duration upper bound from the activity file for the starting activity for the trip to the output Iteration Database. The value is converted from fractional hours to seconds.
SEL_USE_START_END_TIME_LB	If set, directs the Collator to add the end time lower bound from the activity file for the starting activity for the trip to the output Iteration Database. The value is converted from fractional hours to seconds.

Configuration File Key	Description
SEL_USE_START_END_TIME_UB	If set, directs the Collator to add the end time upper bound from the activity file for the starting activity for the trip to the output Iteration Database. The value is converted from fractional hours to seconds.
SEL_USE_START_MODE_PREF	If set, directs the Collator to add the mode preference from the activity file for the starting activity for the trip to the output Iteration Database.
SEL_USE_START_OTHER_PARTICIPANTS	If set, directs the Collator to add the number of other participants from the activity file for the starting activity for the trip to the output Iteration Database.
SEL_USE_START_POSS_LOC	If set, directs the Collator to add the number of possible locations from the activity file for the starting activity for the trip to the output Iteration Database.
SEL_USE_START_PRIORITY	If set, directs the Collator to add the activity priority from the activity file for the ending activity for the trip to the output Iteration Database.
SEL_USE_START_START_TIME_LB	If set, directs the Collator to add the start time lower bound from the activity file for the starting activity for the trip to the output Iteration Database.
SEL_USE_START_START_TIME_UB	If set, directs the Collator to add the start time upper bound from the activity file for the starting activity for the trip to the output Iteration Database.
SEL_USE_START_VEHICLE_ID	If set, directs the Collator to add the vehicle ID from the activity file for the starting activity for the trip to the output Iteration Database.

Appendix C: Iteration Database Microsimulation Event Configuration File Keys

Configuration File Key	Description
SEL_USE_ACCELS	If set, directs the Collator to add the ACCELS field from the Traffic Microsimulator event output data to the output Iteration Database. This value is summed across events from every leg of the trip.
SEL_USE_ANOMALY	If set, directs the Collator to add the ANOMALY field from the Traffic Microsimulator event output data to the output Iteration Database. Note that this gives only the value for the last event of the trip.
SEL_USE_DISTANCE_SUM	If set, directs the Collator to add the DISTANCE_SUM field from the Traffic Microsimulator event output data to the output Iteration Database. This value is summed across events from every leg of the trip.
SEL_USE_EVENT_LEG_ID	If set, directs the Collator to add the LEG field from the Traffic Microsimulator event output data to the output Iteration Database. Only the value for the first event of each trip is reported.
SEL_USE_EVENT_PERSON_ID	If set, directs the Collator to add the TRAVELER field from the Traffic Microsimulator event output data to the output Iteration Database.
SEL_USE_EVENT_TRIP_ID	If set, directs the Collator to add the TRIP field from the Traffic Microsimulator event output data to the output Iteration Database.
SEL_USE_EVENT_USER	If set, directs the Collator to add the USER field from the Traffic Microsimulator event output data to the output Iteration Database.
SEL_USE_EVENT_VEHICLE_ID	If set, directs the Collator to add the VEHICLE field from the Traffic Microsimulator event output data to the output Iteration Database. Only the value for the first event of each trip is reported.
SEL_USE_LINK	If set, directs the Collator to add the LINK field from the Traffic Microsimulator event output data to the output Iteration Database. Note that this gives only the value for the last event of the trip.
SEL_USE_LOCATION	If set, directs the Collator to add the LOCATION field from the Traffic Microsimulator event output data to the output Iteration Database. Note that this gives only the value for the last event of the trip.
SEL_USE_NODE	If set, directs the Collator to add the NODE field from the Traffic Microsimulator event output data to the output Iteration Database. Note that this gives only the value for the last event of the trip.

Configuration File Key	Description
SEL_USE_ROUTE	If set, directs the Collator to add the ROUTE field from the Traffic Microsimulator event output data to the output Iteration Database. Only the value for the first event of each trip is reported.
SEL_USE_SIGNALS	If set, directs the Collator to add the SIGNALS field from the Traffic Microsimulator event output data to the output Iteration Database. This value is summed across events from every leg of the trip.
SEL_USE_STATUS	If set, directs the Collator to add the STATUS field from the Traffic Microsimulator event output data to the output Iteration Database. Note that this gives only the value for the last event of the trip.
SEL_USE_STOPPED	If set, directs the Collator to add the STOPPED field from the Traffic Microsimulator event output data to the output Iteration Database. This value is summed across events from every leg of the trip.
SEL_USE_STOPS	If set, directs the Collator to add the STOPS field from the Traffic Microsimulator event output data to the output Iteration Database. This value is summed across events from every leg of the trip.
SEL_USE_TIME	If set, directs the Collator to add the TIME field from the Traffic Microsimulator event output data to the output Iteration Database. Only the value for the last event of each trip is reported.
SEL_USE_TIME_SUM	If set, directs the Collator to add the TIME_SUM field from the Traffic Microsimulator event output data to the output Iteration Database. This value is summed across events from every leg of the trip.
SEL_USE_TURN	If set, directs the Collator to add the TURN field from the Traffic Microsimulator event output data to the output Iteration Database. Note that this gives only the value for the last event of the trip.
SEL_USE_VEH_SUBTYPE	If set, directs the Collator to add the VSUBTYPE field from the Traffic Microsimulator event output data to the output Iteration Database. Only the value for the first event of each trip is reported.
SEL_USE_VEH_TYPE	If set, directs the Collator to add the VEHTYPE field from the Traffic Microsimulator event output data to the output Iteration Database.
SEL_USE_YIELDS	If set, directs the Collator to add the YIELDS field from the Traffic Microsimulator event output data to the output Iteration Database. This value is summed across events from every leg of the trip.

Appendix D: Iteration Database Router/Plan Configuration File Keys

Configuration File Key	Description
`SEL_USE_COST`	If set, directs the Collator to add the cost from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the sum over all legs.
`SEL_USE_DEP_TIME`	If set, directs the Collator to add the departure time from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the first leg.
`SEL_USE_DRIVER`	If set, directs the Collator to add the driver flag from the Plan file for the trip to the output Iteration Database. The Collator will fill this field with NA, since there are multiple possible values per trip.
`SEL_USE_DURATION`	If set, directs the Collator to add the (expected) duration from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the sum over all legs.
`SEL_USE_END_ACC`	If set, directs the Collator to add the ending accessory ID from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the last leg.
`SEL_USE_END_ACC_TYPE`	If set, directs the Collator to add the ending accessory type from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the last leg.
`SEL_USE_GCF`	If set, directs the Collator to add the generalized cost function from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the sum over all legs.
`SEL_USE_LEG_ID`	If set, directs the Collator to add the leg ID from the Plan file for the trip to the output Iteration Database. The Collator will fill this field with "NA", since there are multiple legs for each trip.

Configuration File Key	Description
`SEL_USE_MAX_TIME`	If set, directs the Collator to add the max time flag from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the last leg.
`SEL_USE_MODE`	If set, directs the Collator to add the mode from the Plan file for the trip to the output Iteration Database. The Collator will fill this field with NA, since there are multiple possible values per trip.
`SEL_USE_PLAN_PERSON_ID`	If set, directs the Collator to add the person ID from the Plan file for the trip to the output Iteration Database.
`SEL_USE_PLAN_USER`	If set, directs the Collator to add the User Field from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the first leg.
`SEL_USE_ROUTER_LAST_OK_ITER`	If set, directs the Collator to add a field containing the number of the last iteration on which no problems were reported for this traveler in the Router's Problem File. A value of `-1` indicates there has been a problem reported on every iteration.
`SEL_USE_START_ACC`	If set, directs the Collator to add the starting accessory from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the first leg.
`SEL_USE_START_ACC_TYPE`	If set, directs the Collator to add the starting accessory type from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the first leg.
`SEL_USE_STOP_TIME`	If set, directs the Collator to add the (expected) arrival time from the Plan file for the trip to the output Iteration Database. Note that there are multiple legs in a trip, each of which contains a value for this field. The Collator will use the last leg.
`SEL_USE_TRIP_ID`	If set, directs the Collator to add the value in the user field from the Plan file for the trip to the output Iteration Database.

Appendix E: Iteration Database Population Configuration File Keys

Configuration File Key	Description
`SEL_USE_<pop_file header field>`	If set, directs the Collator to add the corresponding demographic variable from the population file to the output Iteration Database.
`SEL_USE_BLOCK_GROUP`	If set, directs the Collator to add the `BLOCK_GROUP` field from the population file to the output Iteration Database.
`SEL_USE_HH_ID`	If set, directs the Collator to add the `HH_ID` field from the population file to the output Iteration Database.
`SEL_USE_HOME_LOCATION`	If set, directs the Collator to add the `HOME_LOCATION` field from the population file to the output Iteration Database.
`SEL_USE_NUMBER_PERSONS`	If set, directs the Collator to add the `NUMBER_PERSONS` field from the population file to the output Iteration Database.
`SEL_USE_NUMBER_VEHICLES`	If set, directs the Collator to add the `NUMBER_VEHICLES` field from the population file to the output Iteration Database.
`SEL_USE_PERSON_ID`	If set, directs the Collator to add the `PERSON_ID` field from the population file to the output Iteration Database.
`SEL_USE_TRACT`	If set, directs the Collator to add the `TRACT` field from the population file to the output Iteration Database.

Appendix F: Iteration Database Stratifier Configuration File Keys

Configuration File Key	Description
SEL_ALGORITHM	A comma- or semicolon-separated list of names of algorithms used to select travelers. Currently, the allowed values are: HI_VAR, LO_VAR, HI_RANGE, LO_RANGE, HI_MEAN, LO_MEAN, HI_SIGMA, and LO_SIGMA. These values instruct the Selector to pick the cell with the highest or lowest variance, range, mean, or ratio of mean to standard deviation, respectively. In addition, it is possible to pick a cell by index using the argument PICK_CELL <n>, where <n> is an integer. If the k binnings that make up a cell have n_0, n_1, …, n_k bins each, a set of indexes into each binning of the form (i_0, i_1, …, i_k) is equivalent to the cell index $i_k + n_{(k-1)} * (i_{(k-1)} + … + * (i_2 + n_1 * (i_0))…)$. One algorithm must be supplied for each name in the SEL_COST argument.
SEL_BIN_BOUNDS_n	If specified, the bin boundaries used for binning n. The argument is a comma- or semicolon-separated list of values. Bins will be created extending from the smallest value to - infinity and from the largest value to + infinity. Over-ridden if SEL_BINS_NUMBINS_n is present.
SEL_BIN_FIELD_n	The column name of the input Iteration Database used for creating binning n.
SEL_BIN_NAME_n	The column name for binning n in the output Iteration Database. n must start at 0.
SEL_BIN_NUMBINS_n	If specified, the number of bins to create for binning n. Bin boundaries will be chosen to create bins with equal numbers of elements. The number of bins may be adjusted within the code if the distribution of elements is concentrated on a few values.

Configuration File Key	Description
SEL_BIN_SEL_ALGO	A semicolon-separated list of names of algorithms to use in selecting trips from within the chosen cell of the stratification. Possible values are: RANDOM, TAIL, or ALL. There must be one algorithm supplied for each name in the SEL_COST argument. The RANDOM value takes two optional parameters: the first is the fraction of elements to select; the second is the absolute number of elements to select, which overrides the first if it is smaller. For example: RANDOM, 0.2, 100; TAIL takes one required and three optional arguments. The first is 0 if the lowest cost tail is to be selected and non-zero otherwise; the second is the fraction of trips to select; the third is an absolute threshold to apply; and the last is an absolute number of elements to select.
SEL_COST	A comma- or semicolon-separated list of names of columns in the input Iteration Database to associate with stratifications for use by the Selector in selecting trips. One selected set will be created for each name.
SEL_GOAL	The Activity Regenerator command to be associated with the selected set of trips. One goal must be supplied for each name in the SEL_COST argument. The entire goal string is written to the Activity Generator feedback file after each selected traveler ID.
SEL_STRAT_BINS	A semicolon-separated list of comma-separated strings specifying the names of binnings in the Iteration Database to be used in stratifying the data.
SEL_USE_STRATIFICATION	Each semi-colon separated list in the value of SEL_STRAT_BINS creates one stratification, indexed beginning with 0. One stratification must be supplied for each name in the SEL_COST argument.

Appendix G: Iteration Database Algorithm Configuration File Keys

Configuration File Key	Description
SEL_USE_AND	Directs the Collator to include a field (for each pair) which is the logical AND between the values of the two fields. The argument is a semicolon separated list of comma-separated Iteration Database field names.
SEL_USE_CROSS_BOUND	Adds a field to the output Iteration Database which is true if the starting and ending activity locations for the trip are in different polygons. The argument is a comma-separated list of UAZ ID, as above, but without the polygon identifier. The value of this field is true if the starting and ending activity locations for the trip are in different polygons.
SEL_USE_DIFF	Directs the Collator to include a field (for each pair) which is the difference between the values of the two fields. The argument is a semicolon-separated list of comma-separated Iteration Database field names.
SEL_USE_DRIVES_PASSENGER	Directs the Collator to include a field which is true if, on any leg of the trip, the Plan file specifies that the traveler drives a vehicle with passengers. Does not apply to transit vehicle drivers.
SEL_USE_EFFSPEED	Directs the Collator to include a field giving the ratio of the Euclidean distance between start and end activities to the total time (as calculated for SEL_USE_T_TOTAL).
SEL_USE_END_ACT_USER_DATA	Directs the Collator to include a field giving the value of any user-specified field in the Activity Location network table for the activity location at the end of the trip. The argument is a semicolon-separated list of field names.
SEL_USE_END_IN_REGION	See SEL_USE_START_IN_REGION. The value of this field will be true if the ending activity location is inside the polygon.
SEL_USE_END_REGION	The number (n) of the User Analysis Zone specified by the configuration file key SEL_UAZ_FILE_n. The value of the field in the iteration database is the number of the polygon in the UAZ that contains the ending activity location or –1 if not in any defined polygon.
SEL_USE_EUCLID	Directs the Collator to include a field giving the Euclidean distance between the starting and ending activity locations, in the same units the network tables use.

Configuration File Key	Description
`SEL_USE_FINISH_TRIP`	Directs the Collator to include a field that is true if an "end trip" event is found for this trip in the event output file.
`SEL_USE_MODE_LEG_COUNT`	Directs the Collator to include a field giving the number of legs on the trip using the specified mode. The argument is a comma-separated list of modes. Currently, only the following modes are distinguished: `w` - walk `i` - bicycle `t`, `l`, or `b` - transit `c` - non-transit vehicle (as driver or passenger) `a` - activity
`SEL_USE_MODE_STRING`	Directs the Collator to include a field reflecting the modes used on every leg of this trip. The value is a string with one letter for each leg, starting from the left. Information comes from the Plan file, which does not know about as many modes as the activity file. Currently the letters used and their meanings are: `c` - driving a vehicle `p` - passenger in a non-transit vehicle `t` - transit `w` - walk `i` - bicycle `a` - activity (no transportation)
`SEL_USE_NUMLEGS`	Directs the Collator to include a field giving the number of legs in this trip, as found in the Plan file.
`SEL_USE_OR`	Directs the Collator to include a field (for each pair) which is the logical `OR` between the values of the two fields. The argument is a semicolon-separated list of comma-separated Iteration Database field names.
`SEL_USE_PRODUCT`	Directs the Collator to include a field (for each pair) which is the product of the values of the two fields. The argument is a semicolon-separated list of comma-separated Iteration Database field names.
`SEL_USE_RATIO`	Directs the Collator to include a field (for each pair) which is the ratio between the values of the two fields. The argument is a semicolon-separated list of comma separated Iteration Database field names.

Configuration File Key	Description
SEL_USE_RELDIFF	Directs the Collator to include a field (for each pair) which is the relative difference between the values of the two fields. The argument is a semicolon-separated list of comma-separated Iteration Database field names. SEL_USE_RELDIFF A, B creates a field with values (A - B) / B.
SEL_USE_START_ACT_USER_DATA	Directs the Collator to include a field giving the value of any user-specified field in the Activity Location network table for the activity location at the beginning of the trip. The argument is a semicolon-separated list of field names.
SEL_USE_START_IN_REGION	Adds a field to the Iteration Database which has the value true if the starting activity location is in the specified polygon. The argument is a semicolon-separated list of UAZ region identifiers. Each region identifier is of the form <UAZ_ID>, <polygon id>; where <UAZ_ID> is an integer referring to a User Analysis Zone file specified by the UAZ_FILE_NAME configuration file key and <polygon id> is the ID of a polygon contained within that file. The Iteration Database column will be named START_IN_REGION_<UAZ_ID>_<polygon ID>. Its value will be a boolean, which is true if the starting activity location for the trip is inside the polygon.
SEL_USE_START_REGION	The number (n) of the User Analysis Zone specified by the configuration file key SEL_UAZ_FILE_n. The value of the field in the iteration database is the number of the polygon in the UAZ that contains the starting activity location or –1 if not in any defined polygon.
SEL_USE_SUM	Directs the Collator to include a field (for each pair) which is the sum of the values of the two fields. The argument is a semicolon-separated list of comma-separated Iteration Database field names.
SEL_USE_T_MODE	Directs the Collator to include a field giving the total time (in seconds) spent in a particular mode on this trip, as found from the expected duration of those legs in the Plan file. See SEL_USE_MODE_LEG_COUNT for a list of the modes that can be distinguished. The argument should be a semicolon-separated list of mode characters.

Configuration File Key	Description
SEL_USE_T_TOTAL	Directs the Collator to include a field giving the total time spent on this trip as found from the difference in the TIME field between the last and first events for this trip. The event output file should capture events with "change on trip" status.
SEL_USE_T_WAIT	Directs the Collator to include a field giving the total time spent waiting as found by summing over the times between "begin waiting" and "end waiting" events in the event output file. The event output file must capture these events and must not filter out the TIME field.
SEL_USE_T_WALK	Directs the Collator to include a field giving the total time (in seconds) spent walking on this trip, as found from the expected duration of walk legs in the Plan file.
SEL_USE_TRAVERSE_REGION	The number (n) of the User Analysis Zone specified by the configuration file key SEL_UAZ_FILE_n. The value of the field in the iteration database is true if one of the start/end activity locations is in the specified UAZ and the other is outside the UAZ.
SEL_USE_XOR	Directs the Collator to include a field (for each pair) which is the logical exclusive OR between the values of the two fields. The argument is a semicolon-separated list of comma-separated Iteration Database Field names.

Appendix H: Configuration File

This configuration file creates the database shown in the examples in Section 2. File names have been changed to enhance readability.

```
TRANSIMS_ROOT                /home/Gershwinoutput1/CaseStudy3

# Defaults will be supplied for these activity types if they are not
# defined here. They are used to define tours and subtours.

ACT_HOME_ACTIVITY_TYPE       0
ACT_WORK_ACTIVITY_TYPE       1
ACT_SCHOOL_ACTIVITY_TYPE     7

ACT_ANCHOR_ACTIVITY_TYPE_1   0 # Home
ACT_ANCHOR_ACTIVITY_TYPE_2   1 # Work
ACT_ANCHOR_ACTIVITY_TYPE_3   7 # School
ACT_ANCHOR_ACTIVITY_TYPE_4   8 # College

# The network data, especially activity location data, is required.

NET_DIRECTORY                  /home/.../allstr/network
NET_NODE_TABLE                 Node.tbl
NET_LINK_TABLE                 Link.tbl
NET_POCKET_LANE_TABLE          Pocket_Lane.tbl
NET_PARKING_TABLE              Parking_MICROSIM.tbl
NET_LANE_CONNECTIVITY_TABLE    Lane_Connectivity.tbl
NET_UNSIGNALIZED_NODE_TABLE    Unsignalized_Node.tbl
NET_SIGNALIZED_NODE_TABLE      Signalized_Node.tbl
NET_PHASING_PLAN_TABLE         Phasing_Plan.tbl
NET_TIMING_PLAN_TABLE          Timing_Plan.tbl
NET_SPEED_TABLE                Speed.tbl
NET_LANE_USE_TABLE             Lane_Use.tbl
NET_TRANSIT_STOP_TABLE         Transit_Stop.tbl
NET_SIGNAL_COORDINATOR_TABLE   Signal_Coordinator.tbl
NET_DETECTOR_TABLE             Detector.tbl
NET_TURN_PROHIBITION_TABLE     Turn_Prohibition.tbl
NET_BARRIER_TABLE              Barrier.tbl
NET_ACTIVITY_LOCATION_TABLE    Activity_Location.tbl
NET_PROCESS_LINK_TABLE         Process_Link.tbl

# The Collator pulls together information from
# population, activity, plan, and event files.
# The event file specified here was collected from the
# Traffic Microsimulator with the following configuration file keys:
#   OUT_EVENT_TYPE_3      TRAVELER
#   OUT_EVENT_FILTER_3    STATUS&8; STATUS!&4
#   OUT_EVENT_SUPPRESS_3  ACCELS;LEG;ROUTE;SIGNALS;STOPS;STOPPED;
#   TURN;USER;VEHICLE;VEHTYPE;VSUBTYPE;YIELDS

SEL_POPULATION_FILE          /home/.../pop_converted
SEL_ACTIVITY_FILE            $TRANSIMS_ROOT/activity/AS7
SEL_PLAN_FILE                $TRANSIMS_ROOT/plans/.../plans.RS7
SEL_EVENT_FILE               $TRANSIMS_ROOT/.../CA.output/endtrip
```

```
# The Collator will gather statistics on the entire population unless a
# household file is specified.
# When run in distributed mode, it expects an extension
# like ".tAA" or ".tAB".

SEL_HOUSEHOLD_FILE           /home/.../hh.RS7
SEL_MESSAGE_LEVEL            2
LOG_SELECTOR                 1
LOG_ALL                      1

# UAZ file names start at "1".

SEL_UAZ_FILE_1           $TRANSIMS_ROOT/network/rivers.polygons

MODE_MAP_FILE            $TRANSIMS_ROOT/data/allstr.modes
TRANSIT_ROUTE_FILE       $TRANSIMS_ROOT/network/Transit_Route.tbl
TRANSIT_SCHEDULE_FILE    $TRANSIMS_ROOT/network/Transit_Schedule.tbl

ROUTER_PROBLEM_FILE      /home/.../problems.RS7

SEL_ITDB_FILE            /home/.../itdb.tutorial.tAB
SEL_STRAT_OUT_FILE       /home/.../strat.tutorial

# Creating an index can be time-consuming, and is not required.

SEL_NO_ITDB_INDEX      1

# COLLATOR keys.

SEL_USE_AGE                 1
SEL_USE_START_IN_REGION     1, 1
SEL_USE_END_IN_REGION       1, 2; 1, 3
SEL_USE_START_REGION        1
SEL_USE_END_REGION          1
SEL_USE_CROSS_BOUND         1
SEL_USE_DURATION            1
SEL_USE_T_MODE              w; c

SEL_USE_RATIO      T_MODE_c, T_MODE_w
SEL_USE_AND        START_IN_REGION_1_1, END_IN_REGION_1_2;
START_IN_REGION_1_1, END_IN_REGION_1_3

# STRATIFIER keys

SEL_BIN_NAME_0      duration_bin
SEL_BIN_FIELD_0     DURATION
SEL_BIN_BOUNDS_0    1200

SEL_BIN_NAME_1      1_to_2
SEL_BIN_FIELD_1     AND_START_IN_REGION_1_1_END_IN_REGION_1_2

SEL_BIN_NAME_2      1_to_3
SEL_BIN_FIELD_2     AND_START_IN_REGION_1_1_END_IN_REGION_1_3

SEL_BIN_NAME_3      age_bin
SEL_BIN_FIELD_3     AGE
SEL_BIN_BOUNDS_3    18
```

```
# SELECTOR keys.

# Specify which binnings are used in each n-way table.

SEL_STRAT_BINS      0, 1; 0, 2; 3

# Specify which n-way tables are to be used.
# A table may appear in this list more than once.

SEL_USE_STRATIFICATION   0; 2

# For each selection from a table specified above,
# specify the cell selection algorithm, the associated cost
# function, the within-cell selection algorithm, and the
# command string to be passed to the Activity Regenerator for
# each selected activity

SEL_ALGORITHM           PICK_CELL 3; PICK_CELL 1
SEL_COST                HH; RATIO_T_MODE_c_T_MODE_w
SEL_BIN_SEL_ALGO        ALL; TAIL, 1, , 1.5
SEL_GOAL                LS 0.01; REROUTE

ACT_FEEDBACK_FILE       /home/.../feedback.act
ROUTER_HOUSEHOLD_FILE   /home/.../feedback.router
```

Chapter Six: Index